A Trainable Multi-factored QA System

Radu Ion, Dan Ştefănescu, Alexandru Ceauşu, Dan Tufiş, Elena Irimia, Verginica Barbu-Mititelu
Research Institute for Artificial Intelligence, Romanian Academy
13, Calea 13 Septembrie, Bucharest 050711, Romania
radu@racai.ro, danstef@racai.ro, aceausu@racai.ro, tufis@racai.ro, elena@racai.ro, vergi@racai.ro

Abstract

This paper reports on the construction and testing of a new Question Answering (QA) system, implemented as an workflow which builds on several web services developed at the Research Institute for Artificial Intelligence (RACAI).The evaluation of the system has been independently done by the organizers of the Romanian-Romanian task of the ResPubliQA 2009 exercise. We further describe a principled way of combining different relevance factors used for computing a general score, based on which are returned the paragraphs which are most likely to contain the correct answers to the asked questions. The system was trained on a specific parallel corpus, but its functionality is independent on the linguistic register of the training data. The trained QA system which participated in the ResPubliQA shared task is available as a new web service on the RACAI's web-services page.

ACM Categories and Subject Descriptors: H.3.1 Content Analysis and Indexing – *Indexing Methods*, H.3.1 Content Analysis and Indexing – *Linguistic Processing*, H.3.3 Information Search and Retrieval – *Query formulation*, H.3.3 Information Search and Retrieval – *Search process*, H.3.3 Information Search and Retrieval – *Selection process*, H.3.5 On-line Information Services – *Web-based services*, I.2.7 Natural Language Processing – *Text analysis*

Keywords: Question Answering, question analysis, query formulation, search engine, paragraph selection, paragraph ranking, lexical chains, MERT, web services, workflow.

1. Introduction

This year's ResPubliQA (a subtask of QA@CLEF, website at http://celct.isti.cnr.it/ResPubliQA/) challenge mounted a fresh start up compared to past editions. Instead of searching Wikipedia for the exact minimal answers to the questions, this time the organizers decided on an easier evaluation task, namely that of finding a paragraph of text that contains the answer to a question relevant for the underlying corpus. The corpus of choice is the body of EU legislation, namely a subset of JRC-Acquis (http://wt.jrc.it/lt/Acquis/) that has parallel translations aligned at the paragraph level in Bulgarian, Dutch, English, French, German, Italian, Portuguese, Romanian and Spanish. The organizers went to great lengths to prepare a test bed in which all the participating systems can be objectively evaluated and, very important, *compared to each other*.

Research Institute for Artificial Intelligence (RACAI) participated in this year's Romanian-Romanian ResPubliQA exercise following a methodology that we developed for the last year's system (Ion et al., 2008). Given the requirement specifications for the ResPubliQA exercise we focused on the further development of the last year's snippet[1] selection and ranking module. The most important design decision that we made this year is a MERT-inspired[2] optimization (Och, 2003) of the snippet ranking using a linear combination of snippet relevance (to the user's question) scores.

The philosophy of our QA system has not changed: it's a collection of (classical) operations most of which are implemented as Semantic Web services distributed across our internal network. Each operation in the QA module just performs some sort of data formatting between the various input formats the web services support. More specifically, the system is composed of the following web services:

- **question preprocessing** which is achieved by the TTL web service whose WSDL file is located at http://ws.racai.ro/ttlws.wsdl. In this phase we also classify the input question by calling another web service located at http://shadow.racai.ro/JRCACQCWebService/Service.asmx?WSDL;
- **query generation** from the preprocessed question; there are two different algorithms for generating queries one of which is located at http://shadow.racai.ro/QADWebService/Service.asmx?WSDL and the other one is embedded into the system (with the near-term plan to externalize it);

[1] Last year's QA system had to select appropriate snippets from the Wikipedia documents returned by the search engine. The JRC-Acquis document collection provides the snippet, or paragraph, segmentation thus relieving the system of the task of paragraph composing.

[2] Minimum Error Rate Training

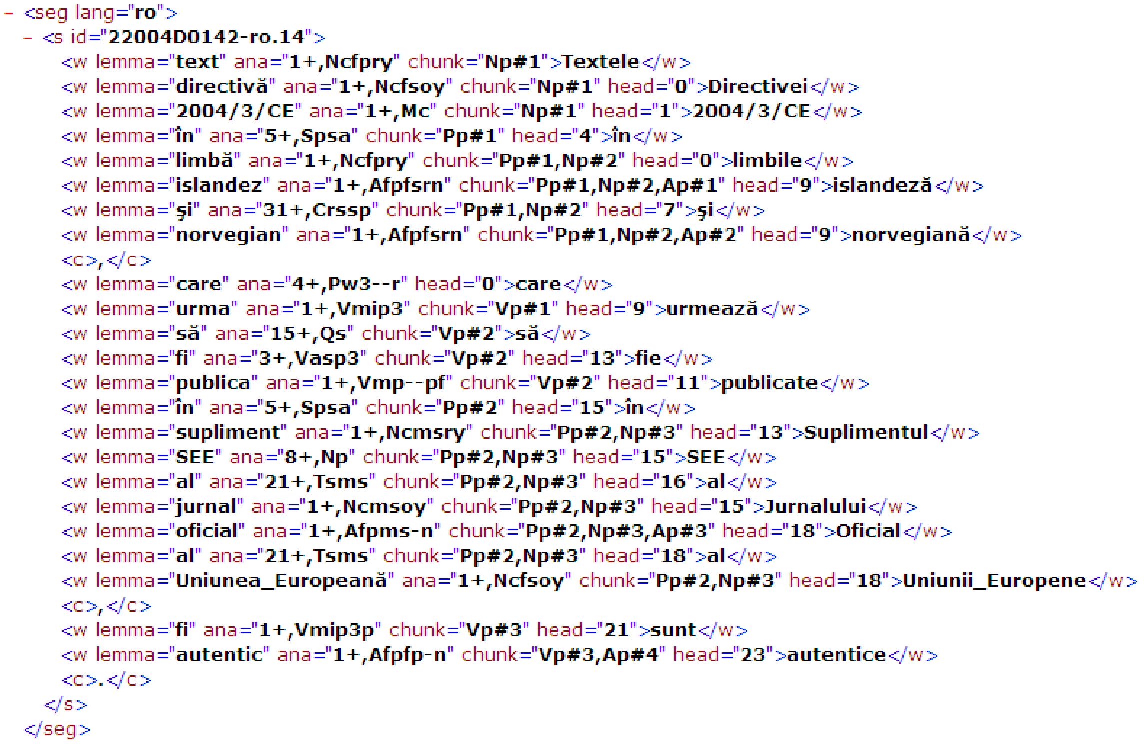
- **search engine querying** which is another web service with its description file at http://www.racai.ro/webservices/search.asmx?WSDL;
- **paragraph ranking** that collects the results from the search engine in the form of a list of relevant paragraphs, computes additional paragraph relevance scores for each paragraph in this list and then linearly combines these scores using a set of MERT-derived weights for each score. The highest scored paragraph will be returned to the user.

Looking back the at the QA track of the Text Analysis Conference in 2008 (TAC 2008, website at http://www.nist.gov/tac/) we see that the statistical component of both document scoring and answer extraction is back in business as in IBM's statistical QA systems in TREC-9 (Ittycheriah et al., 2000) and onwards. This means that, although the QA systems are still very complicated (being basically complex architectures of IR and NLP modules), there are efforts of reducing this complexity in the favor of developing *trainable and principled* QA systems. For instance, Heie et al. (2008) use a language model based approach to sentence retrieval for QA in which every sentence is scored according to the probability P(Q|S) of generating *a specific query* Q (a set of terms) for the respective answer sentence S (also a set of terms). It also is the case that using a linear combination of different relevance measures (probabilities or simply scores) to provide a unique relevance measure of the sentence or paragraph is the de facto choice in recent QA systems (Heie et al., 2008 eq. 3; Wiegand et al., 2008 eq. 2). In what follows, we will present our QA system that makes use of a MERT-like optimization of a linear combination or paragraph relevance scores in order to obtain the paragraph containing the correct answer.

2. Corpus Preprocessing

The corpus to be indexed is a subset of the JRC-Acquis comprising of 10714 documents conforming to the TEI format specifications (http://www.tei-c.org/Guidelines/). That is, roughly, every document has a preamble describing the source of the text along with some other administrative information and a body formed of title, body and annexes[3]. We only took the body of the document into consideration when extracting the text to be indexed. The body part of one JRC-Acquis document is divided into paragraphs, the unit of text that is required by the ResPubliQA task to be returned as the answer to the user's question.

The text has been preprocessed by TTL and LexPar (Tufiş et al., 2008) to achieve the XML format from Figure 1. We performed word segmentation with EUROVOC terminology identification (see the next subsection), POS tagging, lemmatization and dependency linking (although we did not use it in the end).



```
- <seg lang="ro">
  - <s id="22004D0142-ro.14">
      <w lemma="text" ana="1+,Ncfpry" chunk="Np#1">Textele</w>
      <w lemma="directivă" ana="1+,Ncfsoy" chunk="Np#1" head="0">Directivei</w>
      <w lemma="2004/3/CE" ana="1+,Mc" chunk="Np#1" head="1">2004/3/CE</w>
      <w lemma="în" ana="5+,Spsa" chunk="Pp#1" head="4">în</w>
      <w lemma="limbă" ana="1+,Ncfpry" chunk="Pp#1,Np#2" head="0">limbile</w>
      <w lemma="islandez" ana="1+,Afpfsrn" chunk="Pp#1,Np#2,Ap#1" head="9">islandeză</w>
      <w lemma="şi" ana="31+,Crssp" chunk="Pp#1,Np#2" head="7">şi</w>
      <w lemma="norvegian" ana="1+,Afpfsrn" chunk="Pp#1,Np#2,Ap#2" head="9">norvegiană</w>
      <c>,</c>
      <w lemma="care" ana="4+,Pw3--r" head="0">care</w>
      <w lemma="urma" ana="1+,Vmip3" chunk="Vp#1" head="9">urmează</w>
      <w lemma="să" ana="15+,Qs" chunk="Vp#2">să</w>
      <w lemma="fi" ana="3+,Vasp3" chunk="Vp#2" head="13">fie</w>
      <w lemma="publica" ana="1+,Vmp--pf" chunk="Vp#2" head="11">publicate</w>
      <w lemma="în" ana="5+,Spsa" chunk="Pp#2" head="15">în</w>
      <w lemma="supliment" ana="1+,Ncmsry" chunk="Pp#2,Np#3" head="13">Suplimentul</w>
      <w lemma="SEE" ana="8+,Np" chunk="Pp#2,Np#3" head="15">SEE</w>
      <w lemma="al" ana="21+,Tsms" chunk="Pp#2,Np#3" head="16">al</w>
      <w lemma="jurnal" ana="1+,Ncmsoy" chunk="Pp#2,Np#3" head="15">Jurnalului</w>
      <w lemma="oficial" ana="1+,Afpms-n" chunk="Pp#2,Np#3,Ap#3" head="18">Oficial</w>
      <w lemma="al" ana="21+,Tsms" chunk="Pp#2,Np#3" head="18">al</w>
      <w lemma="Uniunea_Europeană" ana="1+,Ncfsoy" chunk="Pp#2,Np#3" head="18">Uniunii_Europene</w>
      <c>,</c>
      <w lemma="fi" ana="1+,Vmip3p" chunk="Vp#3" head="21">sunt</w>
      <w lemma="autentic" ana="1+,Afpfp-n" chunk="Vp#3,Ap#4" head="23">autentice</w>
      <c>.</c>
    </s>
  </seg>
```

Figure 1: Format of the to-be-indexed JRC-Acquis corpus. The sentence ID contains the CELEX code of the document along with the paragraph identifier

[3] This document structure was not always in place. There were cases in which the annex was incorporated into the body or the title was missing or was also incorporated into the body.

2.1 Paragraph classification

The specifications for the ResPubliQA task defines five possible types of questions: “factoid”, “definition”, “procedure”, “reason” and “purpose”. One of the requirements of the QA task is that the answers are to be found in a single paragraph. This requirement and the weak assumption that a paragraph can only answer to one type of question made possible the classification of the paragraphs based on the expected answer type. By labeling the paragraphs with the type of the expected answer we reduced the complexity of the IR problem: given a query, if the query type is correctly identified, the answer is searched through only a portion of the full corpus.

Our approach to paragraph classification is similar to the classification task of sentence selection for summary generation (Kupiec et al., 1995; Luhn, 1959). The classification problem in summary generation is reduced to a function that estimates the probability for a given sentence in a document to be part of the document extract. The features used for summary sentence selection are mainly based on keyword frequencies, sentence length and location.

There are two main differences between the task of paragraph classification and the task of sentence selection for summary generation. The differences are the numbers of classes required for classification and the entities subject to classification. We have five classes and use paragraphs as entities to be classified. The entities considered for the summarization task are the sentences and the classification has only two classes: the sentence is included in the abstract or not. Should the extracted candidates be labeled with classes denoting their role in the rhetorical structure of the document, the summarization task, certainly, will not have only two classes.

The classes used by our paragraph classifier slightly differ from the query types prescribed by the ResPubliQA task specifications. The classes “reason” and “purpose” were merged into a single one because we found that it is very difficult to automatically disambiguate between them for paragraph classification. To improve the precision of paragraph retrieval we added another class: “delete”. This class was used to identify the paragraphs that should not be returned as answer candidates. Among these paragraphs are the titles (for example, “Article 5”), signatures, parts of tables, etc.

To train the paragraph classifier we prepared a collection of around 800 labeled paragraphs. The paragraphs were extracted from the Romanian part of the ResPubliQA corpus. Each paragraph is labeled with one of the classes: “factoid”, “definition”, “procedure”, “reason-purpose” and “delete”. We used 89 labeled paragraphs (unseen during the training phase) to test the precision of the classifier. The paragraphs in the training and test set are not equally distributed among the five classes and their distribution is not the same as in the corpus: we added more paragraphs in the training data for the classes that were more difficult to classify.

We used maximum entropy for paragraph classification. For feature selection we differentiated between clue words, morpho-syntactical, punctuation, orthographical and sentence length related features.

The features are generated for each sentence in the paragraph. To detect the clue words, the classifier takes the first five tokens as features. As morpho-syntactical features, the classifier takes into consideration the morpho-syntactical description of the main verb and whether the sentence has a proper noun or not. As punctuation features are the number of commas, the number of quotes and the final punctuation mark. Another important feature is the number of initial upper cased words. For length related features the classifier uses the length of each sentence and the length of the paragraph (in sentences).

In the classification task of sentence selection for summary generation there is a location feature that accounts for the position of the sentence in the document. For example, if the sentence is in the introduction or in the conclusion it has a high probability of being included in the extract. We did not use the location feature because we it is very difficult to detect the structure of the documents in the ResPubliQA corpus.

The raw evaluation of the classifier precision for each class is presented in Table 1. Although the precision evaluation of the classifier on a test set of only 89 examples has limited statistical confidence, we noticed a considerable increase in search engine precision if the paragraph candidates are filtered using their assigned label. One reasonable explanation is that even when the classification is wrong, both the relevant paragraph and the question have a good chance to be identically labeled. Without filtering the paragraphs by their label it is possible for the answering paragraph not to be among the first 50 search engine hits (this is the number we retain for the further processing).

Label	Precision	Instances
Reason	1	16/16
Definition	0.88	24/27
Procedure	0.83	5/6
Factoid	0.97	33/34
Delete	1	6/6
Total	0.94	84/89

Table 1: Paragraph classification precision (computed on 89 labeled paragraphs)

2.2 Terminology Identification

The JRC-Acquis documents are manually classified using the EUROVOC thesaurus (http://europa.eu/eurovoc/) that has more than 6000 terms hierarchically organized. The version EUROVOC version 4.3 is available for the official 22 UE languages plus Croatian. The EUROVOC thesaurus terms fall into two categories: descriptors and non-descriptors mutually related by means of semantic relationships:

- descriptors, i.e. words or expressions which denote in unambiguous fashion the constituent concepts of the field covered by the thesaurus (e.g. implementation of the law);
- non-descriptors, i.e. words or expressions which in natural language denote the same concept as a descriptor (e.g. application of the law) or equivalent concepts (e.g. enforcement of the law, validity of a law) or concepts regarded as equivalent in the language of the thesaurus (e.g. banana, tropical fruit);
- semantic relationships, i.e. relationships based on meaning, firstly between descriptors and non-descriptors and secondly between descriptors.

The descriptors are organized in 21 top domains (from politics and international relations to ecology, industry or geography) that are represented by micro-thesauri. There are 519 micro-thesauri in which the descriptors are hierarchically organized. The domains and the descriptors have unique identifiers, language independent, through which the multilingual inter-relations can be derived. The English version has 6645 descriptors and the Romanian one has only 4625 descriptors (roughly 70% of the English EUROVOC).

Considering the fact that the technical terms occurring in the JRC-Acquis were supposed to be translated using the EUROVOC terms, the term list of our tokenizer was extended so that these terms be later recognized. If a term is identified it would count as a single lexical token,

As prerequisites for EUROVOC terms identification we had the corpus tokenized, morpho-syntactically annotated, and lemmatized. There were five steps for terminology identification and updating of the language resources backing up our processing tools:

1. The occurrences of the EUROVOC terms were selected from the corpus. In this step only the terms that had the exact occurrence form as the listed EUROVOC term were selected.
2. An inventory of lemma sequence was extracted for each occurrence form.
3. All the occurrences of the lemma sequences were extracted from the corpus.
4. The terms were normalized to their original orthography (mainly case and hyphenation normalization).
5. For each term occurrence was selected a single morpho-syntactical description. Given the fact that most of the terms are noun phrases, the morpho-syntactical description of the head noun was chosen.
6. Each occurrence of the terms received as lemma the EUROVOC descriptor.

For example, the term "*adunare parlamentară*" ("parliamentary assembly") occurred under the inflected forms *"adunarea parlamentară"* ("the parliamentary assembly"), *"adunările parlamentare"* (*"the parliamentary assemblies"*), "*adunărilor parlamentare*" (*"of/to the parliamentary assemblies"*). For these occurrences we used as lemma the listed form of the EUROVOC term that is "*adunare parlamentară*" ("parliamentary assembly") and used the morpho-syntactical description of the head noun "adunare" ("assembly") as the morpho-syntactical description for the occurrence of the term.

3. The QA system

As already stated, the RACAI's QA system is practically a workflow built on top of our NLP web services, retaining as non web service code parts only those sections that have been newly added. It's a trainable system that uses a linear combination of paragraph relevance scores *S*(*p*) to obtain a global relevance (to the question) measure which will be used as the sort key:

$$S(p) = \sum_i \lambda_i s_i, \quad \sum_i \lambda_i = 1 \qquad (1)$$

where s_i is one of the following measures ($s_i \in [0,1]$):

1. an indicator function that is 1 if the question classification is the same as the paragraph classification (let's call this score s_1). We have developed a question classifier that assigns one of the 8 classes (see Section 5) to the input question. Since the paragraph and question classifier have been developed independently, each with its own particularities dictated by the types of text to be classified (questions vs. paragraphs) and performance considerations, we needed a way to map the class of the question onto the class of the

paragraph. Table 2 contains the mapping from question classes to paragraph classes. Thus, s_1 is simply 1 if the guessed class of the question is identical (via the map in Table 2) to that of the candidate paragraph or 0 otherwise;

2. a lexical chains based score computed between lemmas of the candidate paragraph and lemmas of the question (s_2).
3. a BLEU-like (Papineni et al., 2002) score that will reward paragraphs that contain keywords from the question much in the same order as they appear in the question (s_3);
4. the paragraph and document scores as returned by the search engine (s_4 and s_5);

All these measures (except for s_1 which has already been presented) are described in the next sections (see Section 4 and 6).

Question class	Paragraph class
REASON-PURPOSE	Reason
PROCEDURE	Procedure
DEFINITION	Definition
LOCATION	Factoid
NAME	Factoid
NUMERIC	Factoid
TEMPORAL	Factoid
FACTOID	Factoid

Table 2: Mapping from question classes to paragraph classes.

When the QA system receives an input question, it first calls the TTL web service to obtain POS tagging, lemmatization and chunking (see Figure 1 for a depiction of the annotation detail). Then, it calls the question classifier (Section 5) to learn the question class after which two types of queries are computed. Both queries may contain the question class as a search term to be matched with the class of candidate paragraphs. The search engine will return two lists L_1 and L_2 of at most 50 paragraphs that will be sorted according to the eq. 1. The answer is a paragraph *p* from both L_1 and L_2 for which

$$\arg\min_p \left|rank_1(p) - rank_2(p)\right|, \quad rank_1(p) \le K, rank_2(p) \le K, K \le 50 \tag{2}$$

where $rank_1(p)$ is the rank of paragraph *p* in L_1 and $rank_2(p)$ is the rank of paragraph *p* in L_2 and $|x|$ is the absolute value function. Experimenting with different values for K on our 200 questions test set, we determined that the best value for K is 3. When such a common paragraph does not exist, the system returns the NOA string.

Our QA system is trainable in the sense that the weights (λ_i) that we use to combine our paragraph relevance scores are obtained through a MERT-like optimization technique. Since the development question set comprised of only 20 questions, we proceeded to the enlargement of this test set (having the 20 questions as examples). We produced another 180 questions to obtain a new development set of 200 questions simply by randomly selecting documents from the JRC-Acquis corpus and reading them. For each question we provided the paragraph ID of the paragraph that contained the right answer and the question class (one of the classes described in Section 5). The training procedure consisted of:

1. running the QA system on these 200 questions and retaining the first 50 paragraphs for each question according to the paragraph score given by the search engine (s_4);
2. obtaining for each paragraph the set of 5 relevance scores, $s_1 \ldots s_5$;
3. for each combination of λ parameters with $\sum_{i=1}^{5} \lambda_i = 1$ and increment step of 10^{-2}, compute the Mean Reciprocal Rank (MRR) of the 200 question test set by sorting the list of returned paragraphs for each question according to eq. (1);
4. retaining the set of λ parameters for which we obtain the maximum MRR value.

Our systems (each one corresponding to specific algorithm of query generation, see Sections 5 and 7) were individually optimized with no regard to NOA strings. However, since the new ResPubliQA performance measure rewards "I don't know" (NOA) answers over the wrong ones, we added the combination function (eq. 2) in order to estimate the confidence in the chosen answer. At submission time we used a set of λ parameters as resulted from MERT training on the un-discriminated set of questions. We will show that training different sets of parameters for each question class, yields improved results (Section 7).

4. Document and Paragraph Retrieval

For document and paragraph retrieval we used the Lucene search engine (http://lucene.apache.org). Lucene is a Java-based open source toolkit for full-text searching. It has a rich query syntax (http://lucene.apache.org/java/2_3_2/queryparsersyntax.html) allowing to search using Boolean queries, phrase search, term boosting (the importance of the query term is increased) and field-specific term designation (the query term can have as prefix the field name to constrain the search to), etc. Prior to searching, the Lucene search engine requires an index to be build from the document collection. A Lucene index functions like a relational database with only one table. The records of this table are the indexed documents and the document terms are stored into fields. A document field can have two index specific features: indexed and/or stored. The fields can be indexed and/or stored depending on how the data in the field will be used. For example, if we want to retrieve the document unique identification string and the field is only used for display then the terms in the field should only be stored. If we want to retrieve the modified date of the document and search using this date as a query term, the field for the date should be both indexed and stored.

The terms were filtered based on their morpho-syntactic description so that only the content words were indexed. Prior to indexing, the content words were normalized to their lemmatized form. We provide an example of the terms that were indexed for a given sentence in Table 3. There were a few special cases were we had to account for tagging and lemmatization errors. For tokens tagged as proper nouns, foreign words or numerals and for the initial upper-cased tokens or for tokens having a punctuation mark in the middle we indexed the word form on the same position with the lemma. In this way, the term can match if searched both by lemma and word form.

Romanian			English		
Wordform	MSD	Indexed lemma	Wordform	MSD	Indexed lemma
având	Vmg	avea	having	Vmpp	have
în	Spsa		regard_to	Ncns	regard_to
vedere	Ncfsrn	vedere			
Tratatul	Ncmsry	tratat	the	Dd	
			treaty	Ncns	treaty
de	Spsa		establishing	Vmpp	establish
instituire	Ncfsrn	instituire			
a	Tsfs		the	Dd	
Comunității_Europene_a_Energiei_Atomice	Ncfsoy	Comunitatea_Europeană_a_Energiei_Atomice	European_Atomic_Energy_Community	Ncns	European_Atomic_Energy_Community
(			(		
Euratom	Np	Euratom	Euratom	Np	Euratom
)			)		
			,		
			and	Cc-n	
în_special	Rgp	în_special	in	Sp	
			particular	Afp	particular
articolul	Ncmsry	articol	article	Ncns	article
8	Mc	8	8	Mc	8
			thereof	Rmp	thereof
,			;		

Table 3: Terms that are indexed by Lucene

We built two different indexes for the same collection of documents: one where the paragraphs are the index records and one where the documents are the index records. Given a query, the search engine returns a list of paragraphs that best match the query. There are two scores for each paragraph. One is the paragraph score and one is the document score. The two scores represent the score of the paragraph in the paragraph index and the score of the paragraph's document in the document index.

5. Question Preprocessing and Classification. Query Generation

Our QA system's ability to deliver NOA strings is achieved by combining the results of two queries, both of which are automatically derived from the natural language question. In order to do this, the user's question is first preprocessed to obtain an annotation similar to the one in Figure 1. This includes POS tagging, lemmatization and chunking. After that, the question class is derived that will optionally serve as an additional query term that will match the class of the paragraph (see Section 3).

Most QA systems identify the type of the natural language questions inputted by the users. This is called Question Classification (QC) and it is now a standard procedure for labeling the questions according to what they require as answers. It can be achieved in several ways, using rules or automatic classifiers. RACAI experience accumulated during the previous CLEF competitions made us choose the Maximum Entropy (ME) automatic classifier for our QC module. The ME framework proved to be well suited for this task since it can combine diverse forms of contextual information in a principled manner. It is a machine learning technique, originally developed for statistical physics, used for the analysis of the available information in order to determine a unique probability distribution. Similar to any statistical modeling method, it relies on empirical data sample that gives, for some known sets of input, a certain output. The sample is analyzed and a model is created, containing all the rules that could be inferred from the data. The model is then used to predict the output, when supplied with new sets of input that are not in the sample data. The maximum entropy method constructs a model that takes into account all the facts available in the sample data but otherwise preserves as much uncertainty as possible.

In terms of QC, the ResPubliQA task is completely different from the Wikipedia one. The different nature of the two corpora reflects into different type of questions that can be asked. This means we have different question classes for the two corpora and that explains why we couldn't use the QC module built for the competition of the previous year. Therefore, we had to train the ME classifier for the new type of questions. We had 8 classes, namely:

REASON-PURPOSE – usually for Why questions;
PROCEDURE – for questions asking about juridical procedures;
DEFINITION – for questions asking for definitions;
LOCATION – for questions asking about locations;
NAME – for questions asking about named entities: persons, organizations, etc.
NUMERIC – for questions asking for numbers (i.e. *how many…?*);
TEMPORAL – for questions asking about dates;
FACTOID – for factoid questions other than the previous four.

Our problem may be formulated as any classification problem: given a set of training data, one has to find the distribution probability P, such that $P(a|b)$ is the probability of class a given the context b and then apply this probability distribution on unseen data. In our case, the context b is formed by certain features extracted for every question. This year we used the following features extracted after the questions have been preprocessed: the first wh-word in the question, the first main verb, the first noun, the pos-tags for all the nouns, main verbs, adjectives, adverbs and numerals that appear in the question and the first-noun first-verb order. The preprocessing implied POS-tagging and lemmatization of the questions using the existing RACAI tools.

For training, we used the 200 questions test set that we developed starting with the examples of the competition organizers. The model had 100% accuracy on the training data but the figure dropped to 97.2% for the 500 questions in the test data. This means that the classifier assigned the wrong class only for 14 questions out of 500. Most often, the module couldn't disambiguate between FACTOID and PROCEDURE classes. The high score obtained is not a surprise due to the big differences between the features which are characteristic to the chosen classes. This was not an accident. We deliberately chose the classes so that the classifier would not have problems in predicting the correct class. Initially, we had 5 classes, with REASON and PURPOSE being separate classes and LOCATION, NAME, NUMERIC and TEMPORAL being part of the FACTOID class. Due to poor results obtained by the classifier we decided to combine REASON and PURPOSE classes and gradually split FACTOID into those classes enumerated above. We further tried to disambiguate between names of persons and names of organizations but the results were not satisfactory and so we backed off to the presented list of classes.

The first algorithm of query generation (the TFIDF query algorithm) considers all the content words of the question (nouns, verbs, adjectives and adverbs) out of which it constructs a disjunction of terms (which are lemmas of the content words) with the condition that the TFIDF of the given term is above a certain threshold. The TFIDF of each term t of the query has been pre-computed from the entire indexed corpus as

$$TFIDF(t) = \left(1 + \ln(tf)\right) \cdot \ln\left(\frac{D}{df}\right) \tag{3}$$

in which **ln** is the natural logarithm, *tf* is the term frequency in the entire corpus, *df* is the number of documents in which the term appears and *D* is the number of documents in our corpus, namely 10714 (if *tf* is 0, *df* is also 0 and the whole measure is 0 by definition). The rationale behind this decision is that there are certain terms that are very frequent and also very uninformative. As such, they must not be included in the query. Table 4 contains 10 of the most important/unimportant terms with their TFIDF scores. We experimentally set the TFIDF threshold to 9 for Romanian.

Uninformative Term	TFIDF	Informative Term	TFIDF
articol	0.096	emolient	66.890
adopta	0.145	antistatic	56.660
în_special	0.340	emulgator	54.532
consiliu	0.404	biodestructiv	53.967
tratat	0.716	calmant	52.579
comisie	0.719	subsistem	51.952
comunitatea_europeană	0.860	deputat	51.059
stat_membru	1.380	microdischetă	47.868
prevedea	2	opacimetru	46.746

Table 4: Most uninformative and most informative Romanian terms according to the globally computed TFIDF measure

The second algorithm of query generation (QG, the chunk-based query algorithm) also uses the preprocessing of the question. The previous QG module (Ion et al., 2008) we developed for Wikipedia was not working as expected for the new data and so, we had to adapt it to cope with it. As in the previous version, the algorithm takes into account the NP chunks and the main verbs of the question. Now, for each NP chunk, two (instead of one) query terms are constructed: (i) one term is a query expression obtained by concatenating the lemmas of the words in the chunk and having a boost equal to the number of those words, and (ii) the other one is a Boolean query in which all the different lemmas of the words in the chunk are searched in conjunction. For example an "*a b c*" chunk generates the following two queries: `"l(`*a*`) l(`*b*`) l(`*c*`)"^3` and `(l(`*a*`) AND l(`*b*`) AND l(`*c*`))` where l(*w*) is the lemma for the *w* word. For each chunk of length *n*, we generate all the sub-chunks of *n*-1length (i.e. "*a b*" and "*b c*"), and for each such sub-chunk[4] we construct two terms in the fashion just described (for the example above we get `"l(`*a*`) l(`*b*`)"^2, (l(`*a*`) AND l(`*b*`))` for the first sub-chunk, and `"l(`*b*`) l(`*c*`)"^2, (l(`*b*`) AND l(`*c*`))` for the second one). We then continue the procedure until the length of the sub-chunks is 1. Like in the previous version, for each main verb we generate a query term for its lemma but only if it is different from light or generic verbs: fi (*be*), avea (*have*), însemna (*mean*), înțelege (*understand*), întâmpla (*happen*), referi (*refer to*), reprezenta (represent), desemna (*designate*), numi (*name*), define (*define*), considera (*consider*), semnifica (*signify*), denota (*denote*), da (*give*).

The final query is formed as a disjunction of the generated query terms. For example, the question: *"Care este scopul modificării deciziei 77/270/Euratom în ceea ce priveşte energia nucleară?"* is preprocessed into:

```
<seg lang="ro">
<s id="rom.1">
<w lemma="care" ana="Pw3--r">Care</w>
<w lemma="fi" ana="Vmip3s" chunk="Vp#1">este</w>
<w lemma="scop" ana="Ncmsry" chunk="Np#1">scopul</w>
<w lemma="modificare" ana="Ncfsoy" chunk="Np#1">modificării</w>
<w lemma="decizie" ana="Ncfsoy" chunk="Np#1">deciziei</w>
<w lemma="77/270" ana="Mc" chunk="Np#1">77/270</w>
<c>/</c>
<w lemma="Euratom" ana="Np" chunk="Np#2">Euratom</w>
<w lemma="în_ceea_ce_priveşte" ana="Spca" chunk="Pp#1">în_ceea_ce_priveşte</w>
<w lemma="energie_nucleară" ana="Ncfsry" chunk="Pp#1,Np#3">energia_nucleară</w>
<c>?</c>
</s>
</seg>
```

The NP chunks, as recognized by our chunker, are: "*scopul modificării deciziei 77/270*", "*Euratom*" and "*energia nucleară*". The two words in the last chunk have been identified by the preprocessing tool as making a

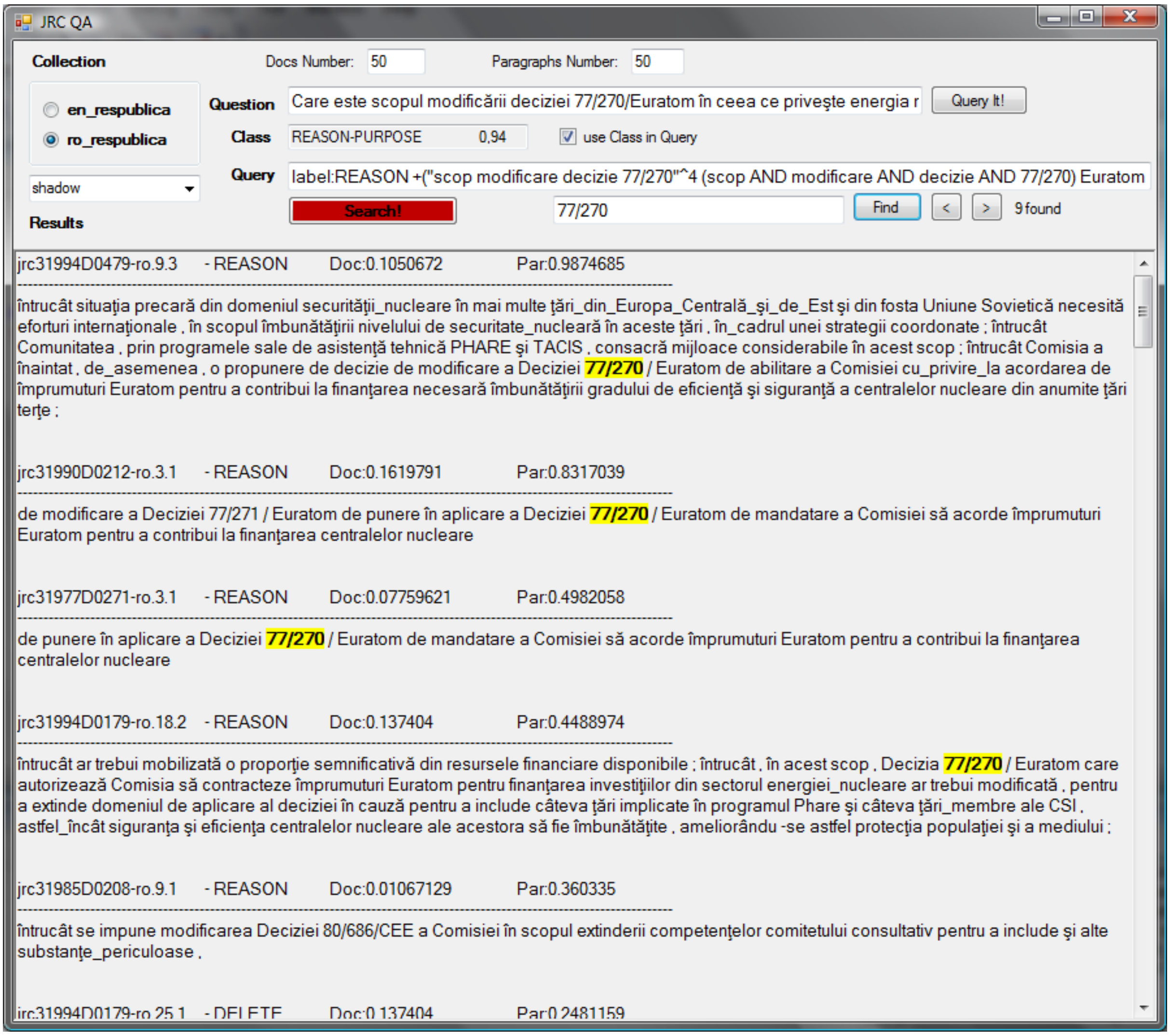
[4] By sub-chunk of a chunk of length *n* we mean a sub-string of it containing *m* words, with *m* < *n*. The sub-string cannot start or end in the middle of a word.

EUROVOC term and they are joined as one lexical token: “*energia_nucleară*”. The main verb is “*este*”, which is in the list of excluded verbs previously described. The query generated starting from these elements is:

```
"scop modificare decizie 77/270"^4 (scop AND modificare AND decizie AND 77/270) Euratom
energie_nucleară "scop modificare decizie"^3 (scop AND modificare AND decizie)
"modificare decizie 77/270"^3 (modificare AND decizie AND 77/270) "scop modificare"^2
(scop AND modificare) "modificare decizie"^2 (modificare AND decizie) "decizie 77/270"^2
(decizie AND 77/270) scop modificare decizie 77/270
```

We set the default Boolean operator between query terms to be OR and so, the previous query is a disjunction. The generation of the query terms aims at maximization of the chances to find the expressions from the questions received as input. The longer the expression, the higher the score of the paragraph that contains it (due to the boost factors). On the other hand, there may be cases when there is no paragraph that contains a certain sequence of words as it appears in the question. In this case, the generated conjunctional queries are supposed to help the engine to find those paragraphs containing as many question keywords as possible. We consider that the top returned paragraphs are those which are the most probable to contain the answers we are looking for.

One frequently used method for enhancing the performance of the QA systems is to enrich the mono-word query terms with synonyms extracted from a lexicon or a lexical ontology like Princeton WordNet. With almost 60,000 synsets, the Romanian WordNet is clearly an excellent resource for such a method. However, due to the fact that the corpus is a specialized one, containing much more technical terms than an ordinary corpus, we chose not to make use of synonyms at the query level but in a subsequent step of computing similarity scores between question and the paragraphs returned as most likely to contain a correct answer. Figure 2 contains a depiction of a C# application that was specially conceived to help us test the performance of the queries.



Docs Number: 50
Paragraphs Number: 50

○ en_respublica
● ro_respublica

shadow

Question: Care este scopul modificării deciziei 77/270/Euratom în ceea ce priveşte energia r

Class: REASON-PURPOSE 0,94 ☑ use Class in Query

Query: label:REASON +("scop modificare decizie 77/270"^4 (scop AND modificare AND decizie AND 77/270) Euratom

77/270 9 found

```
jrc31994D0479-ro.9.3     - REASON     Doc:0.1050672          Par:0.9874685
--------------------------------------------------------------------------------
întrucât situaţia precară din domeniul securităţii_nucleare în mai multe ţări_din_Europa_Centrală_şi_de_Est şi din fosta Uniune Sovietică necesită
eforturi internaţionale , în scopul îmbunătăţirii nivelului de securitate_nucleară în aceste ţări , în_cadrul unei strategii coordonate ; întrucât
Comunitatea , prin programele sale de asistenţă tehnică PHARE şi TACIS , consacră mijloace considerabile în acest scop ; întrucât Comisia a
înaintat , de_asemenea , o propunere de decizie de modificare a Deciziei 77/270 / Euratom de abilitare a Comisiei cu_privire_la acordarea de
împrumuturi Euratom pentru a contribui la finanţarea necesară îmbunătăţirii gradului de eficienţă şi siguranţă a centralelor nucleare din anumite ţări
terţe ;


jrc31990D0212-ro.3.1     - REASON     Doc:0.1619791          Par:0.8317039
--------------------------------------------------------------------------------
de modificare a Deciziei 77/271 / Euratom de punere în aplicare a Deciziei 77/270 / Euratom de mandatare a Comisiei să acorde împrumuturi
Euratom pentru a contribui la finanţarea centralelor nucleare


jrc31977D0271-ro.3.1     - REASON     Doc:0.07759621         Par:0.4982058
--------------------------------------------------------------------------------
de punere în aplicare a Deciziei 77/270 / Euratom de mandatare a Comisiei să acorde împrumuturi Euratom pentru a contribui la finanţarea
centralelor nucleare


jrc31994D0179-ro.18.2    - REASON     Doc:0.137404           Par:0.4488974
--------------------------------------------------------------------------------
întrucât ar trebui mobilizată o proporţie semnificativă din resursele financiare disponibile ; întrucât , în acest scop , Decizia 77/270 / Euratom care
autorizează Comisia să contracteze împrumuturi Euratom pentru finanţarea investiţiilor din sectorul energiei_nucleare ar trebui modificată , pentru
a extinde domeniul de aplicare al deciziei în cauză pentru a include câteva ţări implicate în programul Phare şi câteva ţări_membre ale CSI ,
astfel_încât siguranţa şi eficienţa centralelor nucleare ale acestora să fie îmbunătăţite , ameliorându -se astfel protecţia populaţiei şi a mediului ;


jrc31985D0208-ro.9.1     - REASON     Doc:0.01067129         Par:0.360335
--------------------------------------------------------------------------------
întrucât se impune modificarea Deciziei 80/686/CEE a Comisiei în scopul extinderii competenţelor comitetului consultativ pentru a include şi alte
substanţe_periculoase ,


jrc31994D0179-ro.25.1    - DELETE     Doc:0.137404           Par:0.2481159
```

Figure 2: C# application developed for testing queries

6. Paragraph Relevance Measures

As already stated (in Section 3) our QA system uses a linear combination of paragraph relevance measures (eq. 1) to score a given candidate paragraph as to the possibility of containing the answer to the question. s_1 is a simple score that is 1 if the paragraph has the same class as the question and 0 otherwise. Scores s_4 and s_5 are given by Lucene and measure the relevance of the paragraph to the query (automatically derived from the question). The remaining measures are explained below.

The BLUE-like similarity measure (s_3) between the question and one candidate paragraph stems from the fact that there are questions that are formulated using a percent of words from one paragraph in the order that they appear in the paragraph. For instance, for the question

> *"Care sunt mărfurile pentru care reimportul în aceeaşi stare nu se limitează la mărfurile importate direct din străinătate?"*

and the paragraph that contains the correct answer

> *"Reimportul în aceeaşi stare nu se limitează la mărfurile importate direct din străinătate, ci se aprobă şi pentru mărfurile care au fost plasate sub un alt regim vamal."*

the underlined text is identical, a strong clue that the paragraph contains the correct answer. BLEU (Papineni et al., 2002) is a measure that counts *n*-grams from one candidate translation in one or more reference translations. We use the same principle and count *n*-grams from the question in the candidate paragraph but here is where the similarity to BLEU ends. Our *n*-gram processing:

1. counts only content word *n*-grams (content words are not necessarily adjacent); actually, an *n*-gram is a sliding window of question content word lemmas of a maximum length equal with the length of the question (measured in content words) and a minimum length of 2;
2. each *n*-gram receives a score equal to the sum of TFIDF weights (see Section 5) of its members;
3. the final similarity score is a sum of the scores of question *n*-grams found in a candidate paragraph which is itself reduced to an array of adjacent content words lemmas.
4. to bring this score in the range of 0 … 1, we normalized it in the set of candidate paragraphs returned for one question.

To clarify the method of computing this similarity score, in the example above the question will be transformed into an array of content word lemmas such as {*reimport*, *stare*, *limita*, *marfă*, *importa*, *direct*, *străinătate*}. The candidate paragraph will have the same representation {*reimport*, *stare*, *limita*, *marfă*, *importa*, *direct*, *străinătate*, *aproba*, *marfă*, *plasa*, *regim*, *vamal*}. Now, the longest matching *n*-gram is (*reimport*, *stare*, *limita*, *marfă*, *importa*, *direct*, *străinătate*) and its score is

$$\sum_{t\in\{reimport,stare,\ldots\}} TFIDF(t)$$

The second longest *n*-grams that match are (*reimport*, *stare*, *limita*, *marfă*, *importa*, *direct*) and (*stare*, *limita*, *marfă*, *importa*, *direct*, *străinătate*) with their scores computed in a similar way. The final score will be a sum of all scores of matching *n*-grams.

The Question-Paragraph Similarity measure (QPS measure, s_2) is intended to measure, as the name suggests, the similarity between the question and the paragraphs returned by the search engine as possible answers. It is computed as follows: for a question Q and a candidate paragraph P, we build two lists containing the lemmas corresponding to the content words in the question and to those in the paragraph: LQ and LP. This is a trivial step as both question and paragraph are already preprocessed. We will consider the lemmas in the LQ list as keywords on the basis of which we have to find the most probable paragraph that is related to the question.

The QPS score is the product of two other scores: (i) a lexical similarity score (LSS), (ii) a cohesion score (CS) and (iii) a verb-possible argument pair score (VAS).

(i) The lexical similarity score is computed as follows: each keyword in LQ is compared with those in the LP list, and between every keyword *kq* in LQ and every keyword *kp* in LP lexical similarity score (*SLS*) is computed. If the two keywords are identical, the score is 1. If not, we search for the shortest lexical paths between the two keywords on the semantic relations in the Romanian WordNet. We used the following relations: *synonym*, *hypernym*, *hyponym*, *holo_member*, *derived*, *derivative*, *subevent*, *category_domain*, *holo_portion*, *causes*, *region_domain*, *near_antonym*, *also_see*, *similar_to*, *particle*, *holo_part*, *usage_domain*, *verb_group*, *be_in_state*. We assigned numerical values to every semantic relation in order to be able to compute scores for the lexical paths we found. The strongest relations have the highest values: *synonym* – 1, *hypernym* and *hyponym* – 0.9 while the opposing relation of *near_antonymy* has received the lowest value: 0.4. These values were manually set by linguists on empirical grounds.

The lexical paths resemble very much to what is known in the literature as lexical chains and they can be used in diverse NLP technologies or applications. For example, we also use them for Word Sense Disambiguation (WSD) and in our experiments on this subject, subsequent to the CLEF competition, we automatically computed the values for the semantic relations in the WordNet using the word sense disambiguated corpus SemCor. The new values were completely different from those manually assigned by the linguists. A detailed analysis will be presented in a future paper (Ion and Ştefãnescu, 2009)[5].

The score for a lexical path is computed using the formula:

$$S_{lp} = \frac{3\sum_{i=1}^{l} v(r_i) + 1}{4l} \tag{4}$$

where l is the length of the lexical path, r_i is the i^{th} relation in the lexical path, and $v(r_i)$ is the value assigned to r_i.

We considered that only for S_{lp} scores equal or above 0.8 the lexical similarity between *kq* and *kp* is strong enough to be taken into consideration (and set $SLS = S_{lp}$), otherwise S_{lp} remains 0. If S_{lp} is 0, we should not discard the possibility that we've found some named entities or words that don't appear in the WordNet and so, we compute a surface similarity between them in terms of the Levenshtein Distance (LD), to cover for these cases. The lower the LD score, the grater the surface similarity and therefore, in order to make this score consistent with the lexical similarity measure we used the following formula:

$$S_{ld} = 1 - \frac{ld(kq, kp)}{\max(length(kq), length(kp))} \tag{5}$$

where *ld*(*kq*, *kp*) is the LD score between the two terms, and *length*(*k*) is the character length of the string *k*.

One should notice that the score is now in the [0, 1] interval and, when the two terms are identical the new score is 1.

If S_{ld} is equal or higher than a threshold of 0.5 (the character strings of the two terms are the same for more than the half of their length), then we set $SLS = S_{ld}$.

Now, the lexical similarity score for a question-paragraph pair is computed as:

$$(i) \quad LSS = \frac{\sum_{kq,kp} SLS_{kq,kp}}{T} \tag{6}$$

where T is the total number of situation when $SLS_{kq,kp} \neq 0$.

(ii) The cohesion score is intended to measure the closeness of the question keywords in the paragraphs, irrespective of their order. First all the question keywords are identified in the paragraph and then, the score is computed as the average distance between consecutive indexes. In order to have values in the [0, 1] interval and have the highest score for consecutive keywords, we compute the score like this:

$$(ii) \quad CS = \frac{1}{\frac{\sum_i^n d_i}{n}} = \frac{n}{\sum_i^n d_i} \tag{7}$$

where d_i is the difference between the index positions in the paragraph of the words corresponding to consecutive keywords in the query and n is the total number of such distances. $\frac{\sum_i^n d_i}{n}$ is the average distance between consecutive indexes, mentioned earlier. We have to say that a distance cannot be 0 because between two consecutive words we have a distance of 1.

(iii) For computing verb-possible argument pair score (*VAS*) we consider the pair formed by the main verb of the question and the following noun (or the previous if there is no following one). We consider that there is a high probability to get a verb-argument pair like this. We look for this pair in the paragraph candidate and if we find it, the score is computed as 1 divided by the distance $d_{v\text{-}a}$ between the two elements forming the pair ($d_{v\text{-}a}$ cannot be 0 for no distance can be 0, as explained above). If the pair cannot be found or there is another main verb between the two elements, the score becomes 0.

The final score is computed as a product between the 3 scores: *QPS* = *LSS* * *CS* * *VAS* only for those paragraphs ending with "." or ";". Otherwise, the *QPS* score is 0. The rationale resides in the observation that because of the way the corpus was segmented at the paragraph level, section titles or table fragments were regularly considered paragraphs. However, they are not of any use for our endeavor, and thus, we disregard them.

[5] Our QA system uses now the new obtained values to compute lexical similarities.

We have to mention that we reached *QPS* score formula, after a long series of experiments. In many of them we used different values for the S_{lp} score weights and different thresholds. One variant of *LSS* score formula had, as denominator, the total number of content words in the paragraph instead of *T*. All the combinations were tested on the 200 questions data test bed. In the end, we were mostly satisfied with the one described above.

7. Evaluations

As in every year, QA@CLEF allowed two runs to be submitted for evaluation. Our two runs that have been submitted to ResPubliQA Romanian-Romanian QA contained NOA strings as the accuracy measure proposed by the organizers rewarded systems that instead of giving a wrong answer, gave a "I don't know/I'm not sure" answer:

$$A = \frac{1}{n}\left(n_{AC} + n_U \frac{n_{AC}}{n} \right) \tag{8}$$

where n is the total number of questions (500), n_{AC} is the number of correctly answered questions and n_U is the number of unanswered questions. The assumption is that the system would give the same proportion of correct answers ($\frac{n_{AC}}{n}$) in "a second try" if it were to be run only on the unanswered questions. Since our runs did not contain candidate answers in the case of unanswered questions, thus denying the organizers the possibility to verify this hypothesis, we will provide candidate answers for unanswered questions here and compute the proportion of interest ourselves.

In Section 5 we presented two query formulation algorithms. Each query produces a different set of paragraphs when posed to the search engine thus allowing us to speak of two QA systems. We applied the training procedure described in Section 3 on our 200 questions test set that we developed on each system and ended up with the following values for the λ parameters:

	λ_1	λ_2	λ_3	λ_4	λ_5
The TFIDF query algorithm	0.22	0.1	0.1	0.19	0.39
The chunk query algorithm	0.32	0.14	0.17	0.25	0.12

Table 5: Parameters for paragraph score weighting

With these parameters, each system was presented with the official ResPubliQA 500 questions test set. For each question, each system returned 50 paragraphs that were sorted according to eq. 1 using parameters from Table 5. Table 6 contains the official evaluations[6] of our two runs, ICIA091RORO and ICIA092RORO. The first run corresponds to running the two QA systems with queries exactly as described in Section 5. The second run however, was based on queries that contained the class of the question based on the fact that when we constructed the index of paragraphs (see Sections 2 and 4) we added a field that kept the paragraph class. This tweak brought about a significant improvement in both accuracy and c@1 measure as Table 6 shows.

	ICIA091RORO	**ICIA092RORO**
ANSWERED	393	344
UNANSWERED	107	156
ANSWERED with RIGHT candidate answer	237	260
ANSWERED with WRONG candidate answer	156	84
UNANSWERED with RIGHT candidate answer	0	0
UNANSWERED with WRONG candidate answer	0	0
UNANSWERED with EMPTY candidate answer	107	156
Overall accuracy	**0.47**	**0.52**
Correct answers out of unanswered	0	0
c@1 measure	**0.58**	**0.68**

Table 6: RACAI's official ResPubliQA evaluations on the Romanian-Romanian task.

In order to estimate how many correct answers would have been returned instead of the NOA strings, we ran our QA systems on the 500 questions test set with the exact same settings as per Table 6 obtaining runs ICIA091RORO-NO-NOA and ICIA092RORO-NO-NOA that were combinations resulting from setting K to the

[6] From http://celct.isti.cnr.it/ResPubliQA/ website

maximum allowed value of 50 in eq. 2 (this way, eq. 2 always returned a paragraph thus eliminating the presence of the NOA string). Then, for the two pairs of runs (ICIA091RORO, ICIA091RORO-NO-NOA) and (ICIA092RORO, ICIA092RORO-NO-NOA) we checked the status of every NOA string by seeing if the corresponding answer was correct or not (using the official Gold Standard of Answers of the Romanian-Romanian ResPubliQA task that we got from the organizers). Table 7 contains the results.

	ICIA091RORO	**ICIA092RORO**
ANSWERED	393	344
UNANSWERED	107	156
ANSWERED with RIGHT candidate answer	237	260
ANSWERED with WRONG candidate answer	156	84
UNANSWERED with RIGHT candidate answer	**32**	**21**
UNANSWERED with WRONG candidate answer	**75**	**135**
UNANSWERED with EMPTY candidate answer	**0**	**0**
Overall accuracy	0.47	0.52
Predicted correct answers/Actual correct answers	**50/32**	**81/21**
c@1 measure	0.58	0.68
Reevaluated overall accuracy	**0.54**	**0.56**

Table 7: RACAI's reevaluated results when NOA strings have been replaced by actual answers

It shows that the assumption that the organizers made does not hold in our case. For ICIA091RORO there is a 4% difference between the c@1 measure and the reevaluated overall accuracy and 12% difference between the same measures in the case of ICIA092RORO. This observation supports the idea that QA systems should be evaluated by the exhibited ability of finding the right information on the spot and possibly in real time.

We ended Section 3 with the hypothesis that training different sets of λ parameters for each class would result in better performance (overall accuracy). We experimented with our 200 questions test set and trained different sets of parameters (with the increment step of 0.05 to reduce the time complexity) for each paragraph class since the class identity (between question and paragraph) is made at the paragraph level. Table 8 presents the trained values for the parameters.

		λ_1	λ_2	λ_3	λ_4	λ_5
The TFIDF query algorithm	Factoid	0.1	0	0.2	**0.4**	0.3
	Definition	**0.2**	0.15	0.05	0.15	0.45
	Reason	0.1	0	0.15	0.3	0.45
	Procedure	0.1	0	0.15	0.15	**0.6**
The chunk query algorithm	Factoid	0.15	0	0.3	0.3	0.25
	Definition	0.05	**0.5**	0.15	0.1	0.2
	Reason	**0.2**	0	**0.4**	0.2	0.2
	Procedure	0.15	0.1	0.25	0.2	0.3

Table 8: Different parameters trained for different classes

Running our QA systems onto the 500 questions test set, sorting the returned paragraphs for each question using the set of parameters trained onto the question's class and combining the results with K = 50 to remove the NOA strings, we obtained an overall accuracy of **0.5774**, with almost 2% better than our best result. This confirmed our hypothesis that class-trained parameters would improve the performance of the system.

8. Conclusions

The CLEF campaign has gone long way into the realm of Information Retrieval and Extraction. Each year, the competition showed its participants how to test and then, how to improve their systems. The competitive framework has motivated systems' designers to adopt daring solutions and to experiment in order to obtain the best result. However, we should not forget that we are building these IR and QA systems primarily to help people to search for the information they need. In this regard, it would be very helpful that future competitions would require that, for instance the QA systems to be available on the Internet. Or, to impose a very strict time limit (one day for instance). Another dimension in which we can extend the usefulness of the competition is the scoring formulas. The requirement that only the first answer is to be evaluated is a little harsh given the willingness of the average user to inspect, say at least 5 top returned results.

We want to extend the present QA system so that it can cross-lingually answer question form English or Romanian in either English or Romanian. We have already processed the English side of the JRC-Acquis and, given that we have several functional Example-Based and Statistical Machine Translation Systems, we may begin by automatically translating either the natural language question or the generated query. Then the combination method expressed by equation 2 would probably yield better results if applied on English and Romanian paragraph lists since a common paragraph means the same information found via two different languages.

The principal advantage of this approach to QA is that one has an easily extensible and trainable QA system. If there is another way to asses the relevance of a paragraph to a given question, simply add another parameter that will account for the importance of that measure and retrain. We believe that in the interest of usability, understandability, adaptability to other languages and, ultimately progress, such principled methods are to be preferred over the probably more accurate but otherwise almost impossible to reproduce methods.

References

M.H. Heie, E.W.D. Whittaker, J.R. Novak, J. Mrozinski and S. Furu. 2008. *TAC2008 Question Answering Experiments at Tokyo Institute of Technology*. In Proceedings of the Text Analysis Conference (TAC 2008), National Institute of Standards and Technology, Gaithersburg, Maryland, USA, November 17-19, 2008.

R. Ion, D. Ştefănescu, Al. Ceauşu and D. Tufiş. 2008. *RACAI's QA System at the Romanian-Romanian Multiple Language Question Answering (QA@CLEF2008) Main Task.* In Alessandro Nardi and Carol Peters (eds.), Working Notes for the CLEF 2008 Workshop, 10 p., Aarhus, Denmark, September 2008.

R. Ion and D. Ştefănescu. 2009. *Unsupervised Word Sense Disambiguation with Lexical Chains and Graph-based Context Formalization*. Submitted to 4th Language & Technology Conference: Human Language Technologies as a Challenge for Computer Science and Linguistics, Poznań, Poland, November 6-8, 2009.

A. Ittycheriah, M. Franz, W-J Zhu, A. Ratnaparkhi and R.J. Mammone. 2000. *IBM's Statistical Question Answering System*. In Proceedings of the 9th Text Retrieval Conference (TREC-9), pp. 229—235, Gaithersburg, Maryland, November 13-16, 2000.

J. Kupiec, J. Pedersen and F. Chen. 1995. *A Trainable Document Summarizer*. In Proceedings of the 18th Annual Int. ACM/SIGIR Conference of Research and Development in IR, pp 68-73, Seattle, WA, July 1995.

H.P. Luhn. 1959. The automatic creation of literature abstracts. IBM J. Res. Develop., 2:159-165, 1959.

F.J. Och. 2003. *Minimum Error Rate Training in statistical machine translation*, In Proceedings of the 41st Annual Meeting on Association for Computational Linguistics, pp.160—167, Sapporo, Japan, July 07-12, 2003.

K. Papineni, S. Roukos, T. Ward and W. J. Zhu. 2002. *BLEU: a method for automatic evaluation of machine translation*. In Proceedings of the ACL-2002: 40th Annual meeting of the Association for Computational Linguistics, pp. 311–318, Philadelphia, July 2002.

D. Tufiş, R. Ion, Al. Ceauşu and D. Ştefănescu. 2008. *RACAI's Linguistic Web Services.* In Proceedings of the 6th Language Resources and Evaluation Conference - LREC 2008, Marrakech, Morocco, May 2008. ELRA - European Language Ressources Association. ISBN 2-9517408-4-0.

M. Wiegand, S. Momtazi, S. Kazalski, F. Xu, G. Chrupała and D. Klakow. 2008. *The Alyssa System at TAC QA 2008*. In Proceedings of the Text Analysis Conference (TAC 2008), National Institute of Standards and Technology, Gaithersburg, Maryland, USA, November 17-19, 2008.